Asfalia

Security Audit

Kabosu

Asfalia Audit on March 28, 2023

Table of Contents

Summary

This report has been prepared for Kabosu to discover issues and vulnerabilities in the source code of the Kabosu project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilising Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from Medium to informational.
We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Kabosu
Platform	EVM
Chain	Ethereum Mainnet
Language	Solidity
Codebase	Files provided
Commit	v0.8.15+commit.e14f2714

Audit Summary

Delivery Date	28/03/2023
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Excellence Security Scoring: 92 / 100

Total	7
Pending	0
Acknowledged	7
Unresolved	0
Partially Resolved	0
Resolved	0

● Critical	● High	● Medium	● Low	○ Informational	● Optimization
0	1	0	1	3	2

Scope

Repository:	N/A
Technical Documentation:	N/A
Contracts:	kabosu.sol

Project Overview

Kabosu is a meme token base off:
Kabosu, the female Shiba Inu featured in the original meme, was a pedigree puppy who was sent to an animal shelter when her puppy mill shut down.

She was adopted in 2008 by Japanese kindergarten teacher Atsuko Satō, and named after the citrus fruit kabosu because Sato thought she had a round face like the fruit.
Kabosu was first pictured in a 2010 blog post by Sato; afterwards, variations of the pictures using overlaid Comic Sans text were posted from a Tumblr blog, Shiba Confessions. However, the use of the intentionally misspelled "doge" dates back to January 2009, when it was mentioned in an episode of Homestar Runners's puppet series.

Project Architecture & Fee Models

1% Buy Tax - 1% Sell Tax

Contract Dependencies

N/A

Privileged Roles

N/A

Findings

Total Issues:
7

● Critical	● High	● Medium	● Low	● Informational	● Optimization
0	1	0	1	3	2

ID	Title	Type	Categories	Severity	Status
#1	Unchecked Call Return Value	SWC-104	Coding Style	Low	Acknowledged
#2	Broken Code	**Custom**	Coding Style	Informational	Acknowledged
#3	Write After Write	**Custom**	Coding Style	Informational	Acknowledged
#4	Missing Event	**Custom**	Coding Style	Informational	Acknowledged
#5	Code With No Effects	SWC-135	Coding Style	Optimization	Acknowledged
#6	Code With No Effects	SWC-135	Coding Style	Optimization	Acknowledged
#7	Centralization	**Custom**	Centralization / Privilege	High	Acknowledged

#1 SWC-104 Unchecked Call Return Value

Category	Severity	Location	Status
Coding Style	Low	Line 649-662	Acknowledged

Description

Unused return value for the addLiquidityETH function call.

Recommendation

Should check the return value of addLiquidityETH is true to ensure liquidity is being added correctly.

Alleviation

N/A

#2 Custom Broken Code

Category	Severity	Location	Status
Coding Style	Informational	Line 409-513	Acknowledged

Description

Function returnToNormalTax reverts, as it is trying to set the buyOperationsFee to 20. This makes the buyTotalFees to be 20, which is greater than the value of 15 specified as the max on L512. Currently no impact as this function performs the same purpose that updateBuyFees and updateSellFees can perform.

Recommendation

Function should be revised. The comments on the require statements state that sellTotalFees must be kept at 30% or less and buyTotalFees must be kept at 15% or less, which conflicts with the require statement comments in the functions updateBuyFees (L487) and updateSellFees (L496).

Alleviation

N/A

#3 Custom Write After Write

Category	Severity	Location	Status
Coding Style	Informational	Line 704 & 706	Acknowledged

Description
The bool `success` is declared and then defined twice for two calls.

Recommendation
Different bools should be used for each different check on a call performing correctly.

Alleviation

N/A

#4 Custom Missing Event

Category	Severity	Location	Status
Coding Style	Informational	Line 438-443, 490-497	Acknowledged

Description
Functions missing an event.

Recommendation
Add events similar to the ones implemented in other onlyOwner restricted functions.

Alleviation

N/A

#5 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Optimization	Line -//-	Acknowledged

Description

There are many lines of code that are used for detecting bots/snipers as well as for placing restrictions on early trading of the token on launch. Given that this contract is for a migration and not a new launch, much of this code can be removed to lower the gas impact on users interacting with the contract.

Recommendation

Recommend removing code with no effect for contract optimization.

Alleviation

N/A

#6 SWC-135 Code With No Effects

Category	Severity	Location	Status
Coding Style	Optimization	Line 255, 318, 354, 431-435, 549, 556	Acknowledged

Description

The variable maxWalletAmount serves no purpose, given that individuals can create multiple addresses to bypass this with ease.

Recommendation

Recommend removing code with no effect for contract optimization.

Alleviation

N/A

#7 Custom Centralization

Category	Severity	Location	Status
Centralization / Privilege	High	Line 404-406 & 408-412	Acknowledged

Description

Functions manageBoughtEarly and massManageBoughtEarly allow for the owner role to manually declare an address as true or false on the boughtEarly array. This makes an address only able to transfer tokens to the owner or burn address.

Recommendation

Recommend removing this function.

Alleviation

N/A

General Comments

- Overall, code is quite clean but could massively benefit from streamlining it by removing code that is no longer necessary given it is undergoing a migration and not needing to deal with all the code needed to handle a smooth launch.

- Recommend adding a function for recovering stuck tokens sent to the contract, similar to the withdrawStuckETH function on L718. Note that this would need to not allow for recovering Kabosu tokens as these are stored on the contract as part of its fee structure.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrectoperations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Block Timestamp

Be aware that the timestamp of the block can be manipulated by a miner.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Asfalia’s prior written consent in each instance.This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Asfalia to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-freenature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. Asfalia’s position is that each company and individual are responsible for their own due diligence and continuous security. Asfalia’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Asfalia is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Project is potentially vulnerable to 3rd party failures of service - namely in the form of APIs providing the price for the currencies used by the project. Project could become at risk if these APIs provided incorrect pricing.

Audit does not claim to address any off-chain functions utilized by the project.

Asfalia

The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone.

With over 30 years of combined experience in the DeFi space, our team is highly dedicated to delivering a product that is as streamlined and secure as possible. Our mission is to set a new standard for security in the auditing sector, while increasing accessibility to top tier audits for all projects in the crypto space. Our dedication and passion to continuously improve the DeFi space is second to none.